High Performance Computing | Custom Modules

After installing an application, variables such as `PATH` and `LD_LIBRARY_PATH` must be set in order to call the application from any directory. The preferred method of setting the environment is to use the `module load` command. Typing `module avail` shows the modules installed by staff that are available system wide.

Users may also create their own custom modules.

Creating a custom module

Here is the general procedure to create a module for user maintained software.

Make a directory for the module files:
mkdir /usr/local/usrapps/$GROUP/modulefiles

In the ***modulefiles*** directory, make a directory for each software, e.g.,
mkdir /usr/local/usrapps/$GROUP/modulefiles/appname

In the directory ***appname***, create a text file appropriately named to indicate version number and, if applicable, the compiler version. The staff naming convention for modules is generally ***appname/[app version number]-[name and version of compiler]***

Module files start with the line
#%Module

followed by definitions for the environment variables. To see examples of how to define the environment variables, see the contents of various staff created modules here: ***/usr/local/apps/modulefiles***. For an example of a more complex module file, see one of the Intel modules, e.g., ***/opt/intel/modulefiles/intel/2017.1.132***.

The default module can be set by creating a text file called ***.version*** in the directory ***/usr/local/usrapps/$GROUP/modulefiles/appname/*** containing
#%Module
set ModulesVersion [app version number]-[name and version of compiler]

Loading a custom module

To load the module, the whole path may be specified

module load /usr/local/usrapps/$GROUP/modulefiles/appname/[app version number]-[name and version of compiler]

Alternatively, the path to the modulefiles can be added by doing the following. (Do not add a slash '/' after ***modulefiles***.)

module use --append /usr/local/usrapps/$GROUP/modulefiles

After doing `module use --append`, the custom modules should be visible when typing `module avail`, and then modules may be loaded as usual:

module load appname

Example

Here is a full example of creating a module. This was done to install and create a module for SPAdes in the software group **bioinfo**. (Only members of **bioinfo** can access this module.)

Install the application

The following was done to install SPAdes:

mkdir /usr/local/usrapps/bioinfo/spades
cd /usr/local/usrapps/bioinfo/spades
wget http://cab.spbu.ru/files/release3.14.0/SPAdes-3.14.0-Linux.tar.gz
tar -xzf SPAdes-3.14.0-Linux.tar.gz
rm SPAdes-3.14.0-Linux.tar.gz

This results in a directory called ***SPAdes-3.14.0-Linux***, containing the directories

bin #contains the executables
share #often contains documentation

Create the module

To create the module, first create the ***modulefiles*** and ***appname*** directories:

mkdir /usr/local/usrapps/bioinfo/modulefiles
cd /usr/local/usrapps/bioinfo/modulefiles
mkdir spades
cd spades

SPAdes was downloaded as a binary executable rather than being compiled; therefore, the naming convention is ***appname/[app version number]***. The version number was **3.14.0**.

From the directory ***/usr/local/usrapps/bioinfo/modulefiles/spades***, create a text file called ***3.14.0*** containing

#%Module
prepend-path PATH {/usr/local/usrapps/bioinfo/spades/SPAdes-3.14.0-Linux/bin};

Set the default module

In case there are or will be other versions of SPAdes installed, set **3.14.0** as default by creating a

file ***/usr/local/usrapps/bioinfo/modulefiles/spades/.version*** containing

```
#%Module
set ModulesVersion 3.14.0
```

Call the module

The module for SPAdes may be called from the command line or from an LSF batch script by doing

```
module load /usr/local/usrapps/bioinfo/modulefiles/spades/3.14.0
```

or by doing

```
module use --append /usr/local/usrapps/bioinfo/modulefiles
module load spades
```

Set the modules path to be available on login

To make the modules available upon logging in, without having to specify the full path or doing `module use`, add the following to the ***~/.tcsh*** or ***~/.bashrc*** file:

```
#User defined modules
module use --append /usr/local/usrapps/bioinfo/modulefiles
```

Usually modifications to the login files (.tcsh, .bashrc., .login) are discouraged, but this an appropriate use case for modifying those files.